Import java.io.\*;

Import java.util.\*;

Public class Autocorrector {

Trie trie = new Trie();

Map<String, Integer> dict = new HashMap<>();

Final static List<String> invalid = Arrays.asList(“abcdefghijklmnopqrstuvwxyz”);

//Time O(S), Space O(S) S is the number of words in dictionary

Public void useDictionary(String dictionaryFileName) throws IOException {

Try {

FileReader fr = new FileReader(dictionaryFileName);

BufferedReader br = new BufferedReader(fr);

String line = null;

While ((line = br.readLine()) != null) {

String word = line.toLowerCase();

If (!line.contains(“ “)) {

Dict.put(word, dict.getOrDefault(word, 0)+1);

Trie.add(word);

} else {

String[] strs= line.split(<\\s>);

For(String str: strs) {

Dict.put(str, dict.getOrDefault(str, 0)+1);

Trie.add(str);

}

}

}

Fr.close();

Br.close();

} catch (FileNotFoundException e) {

System.err.println€;

} catch (IOException e) {

System.err.println€;

}

}

//Time O(S\*n\*m) ~ O(S), Space(K\*N), S is number of words in dictionary

//K is number of edit distance, N is the max number of similar words

Public String suggestSimilarWord(String inputWord) {

If (inputWord.length()==0 || inputWord==null || invalid.contains(inputWord.toLowerCase()) )

Return null;

String s = inputWord.toLowerCase();

String res=null;

TreeMap<Integer, TreeMap<Integer, TreeSet<String>>> map = new TreeMap<>();

TrieNode node = trie.find(s);

If(node == null) {

For (String w: dict.keySet()) {

Int dist = editDistance(w, s);

TreeMap<Integer, TreeSet<String>> similarWords = map.getOrDefault(dist, new TreeMap<>());

Int freq = dict.get(w);

TreeSet<String> set = similarWords.getOrDefault(freq, new TreeSet<>());

Set.add(w);

similarWords.put(freq, set);

map.put(dist, similarWords);

}

Res = map.firstEntry().getValue().lastEntry().getValue().first();

} else if (node !=null) {

Res = s;

}

Return res;

}

//Time O(n\*m), Space O(n\*m), n is word1 length, m is word2 length

Private int editDistance(String word1, String word2) {

Int n = word1.length();

Int m = word2.length();

Int dp[][] = new int[n+1][m+1];

For (int I = 0; I <= n; i++) {

For (int j = 0; j <= m; j++) {

If (I == 0)

Dp[i][j] = j;

Else if (j == 0)

Dp[i][j] = I;

Else if (word1.charAt(i-1) == word2.charAt(j-1))

Dp[i][j] = dp[i-1][j-1];

Else if (i>1 && j>1 && word1.charAt(i-1) == word2.charAt(j-2)

&& word1.charAt(i-2) == word2.charAt(j-1))

Dp[i][j] = 1+Math.min(Math.min(dp[i-2][j-2], dp[i-1][j]), Math.min(dp[i][j-1], dp[i-1][j-1]));

Else

Dp[i][j] = 1 + Math.min(dp[i][j-1], Math.min(dp[i-1][j], dp[i-1][j-1]));

}

}

Return dp[n][m];

}

Public static void main(String[] args) throws IOException {

Autocorrector sc = new Autocorrector();

Sc.useDictionary(“dictionary.txt”);

System.out.println(“edit distance:”);

System.out.println(“bird -> ird: “ + sc.editDistance(“bird”, “ird”));

System.out.println(“ohuse -> house: “ + sc.editDistance(“ohuse”, “house”));

System.out.println(“zopper -> top: “ + sc.editDistance(“zopper”, “top”));

System.out.println(“ask -> askhim: “ + sc.editDistance(“ask”, “askhim”));

System.out.println(“suggest of aop is “ + sc.suggestSimilarWord(“aop”));

System.out.println(“suggest of bloat is “ + sc.suggestSimilarWord(“bloat”));

System.out.println(“suggest of reah is “ + sc.suggestSimilarWord(“reah”));

}

}